John Zank

Project 3 Pseudocode

IO Class

1. biuldList(LinkedList)

```
WHILE iInput is not -1 or the list is empty
    TRY
        Print “Enter an integer”; input is stored into iInput
        if iInput is not -1
            insert iInput to back of list
            increment nodeCount by 1
    CATCH(invalid input)
        Print “ Error: …”
return the list
```

2. displayMenu(LinkedList)

```
Do
    Print (Menu selection items)
    TRY
    Store input of menu selection in choice variable
    SWITCH(CHOICE)
        Case 1: // Delete First
            Print”You have deleted first item” and display value
            delete first item
            Ask user if they wish to continue
            WHILE input is not “y” or “n”
                Print “Invalid choice”; Ask for input again
            IF input is “n”
                choice = 8
            break;
        Case 2: // Delete Last
            Print”You have deleted the last item” and display value
            delete last item
            Ask user if they wish to continue
            WHILE input is not “y” or “n”
                Print “Invalid choice”; Ask for input again
            IF input is “n”
                choice = 8
            break;
        Case 3: // Insert First
            TRY
                Ask user what they wish to insert to front of list
                IF input is less than or equal to the first link’s value
                    Print input has been inserted to front of list
                    Insert input to the front
                ELSE
                    Print input is larger than the first item, will  not insert
                Ask user if they wish to continue
                WHILE input is not “y” or “n”
```

```
                    Print “Invalid choice”; Ask for input again
                IF input is “n”
                    choice = 8
                break;
            CATCH(invalid input)
                Print “Error…”
Case 4: // Insert Last
        TRY
                Ask user what they wish to insert to the end
                IF input is less than or equal to the first link’s value
                    Print input has been inserted to front of list
                    Insert input to the front
                ELSE
                    Print input is larger than the first item, will  not insert
                WHILE input is not “y” or “n”
                    Print “Invalid choice”; Ask for input again
                IF input is “n”
                    choice = 8
                break;
            CATCH(invalid input)
                Print “Error….”
Case 5: // Delete any arbitrary node
        TRY
                Ask user the value of the node they wish to delete
                IF(input is found)
                    Print “Deleting node”
                    Delete searched node
                Else
                    Print “Node not found”
                Ask user if they wish to continue
                WHILE input is not “y” or “n”
                    Print “Invalid choice”; Ask for input again
                IF input is “n”
                    choice = 8
                break;
            CATCH(invalid input)
                Print “Error…”
Case 6: // Insert node after
        TRY
                Ask user for value of node they wish to insert after
                store input in key variable
                Ask user for value they wish to insert
                store into iInput
                Boolean insert = insertAfter method
                IF(insert is true)
                    Print “input has been stored”
                ELSE
                    Print “Key was not found, value not stored”
                Ask user if they wish to continue
```

```
                                    WHILE input is not “y” or “n”
                                        Print “Invalid choice”; Ask for input again
                                    IF input is “n”
                                        choice = 8
                                    break;
                                CATCH(invalid input)
                                    Print “Error…”
                            Case 7: // Display updated list
                                Print “ Updated Linked List”
                                display linked list
                                break;
                            Case 8: // Terminate
                                Print “Exiting program. Goodbye.”
                                System.exit(0)
                            default:
                                Print “Invalid Option.”
                                break;
                        CATCH(invalid input)
                            Print “Error…”
                WHILE(choice is not 8)
```

3. getNodeCount()

```
return nodeCount
```

Link Class

1. getData()

```
return iData;
```

2. getNext()

```
return next;
```

3. getPrevious()

```
return previous;
```

4. setNext(Link)

```
next = lNext;
```

5. setPrevious(Link)

```
previous = lPrevious;
```

John Zank

LinkedList Class

1. isEmpty

check to see if first is null and return boolean result

2. insertFirst(int)

create a new link
IF(linked list is empty)
    last position is the new link
ELSE
    set first positions previous variable equal to the new links location
    set new link's next to first

3. insertLast(int)

create a new link
IF(linked list is empty)
    first position is the new link
ELSE
    set last positions next variable to new link
    set new link's previous to last

4. insertAfter(int, int)

create link Current and equal first
WHILE(Current data is not equal to the key)
    Current = next
    IF(Current equals null)
        return false; key not found
create new link
IF(Current equals last)
    set new link's next to null
    last = new link
ELSE
    set new link's next to Current's next
    set current's next to the previous
set new link's previous to current
set current's next to new link

5. deleteFirst()

create link Temp and equal first
IF(first's next is equal to null)
    last equals null
ELSE
    Set first's next's previous to null
first equals first's next
return Temp

6. deleteLast()

create link temp and equal last
IF(first's next is equal to null)

last equals null
ELSE
first's next's previous is null
first equals first's next
return Temp

7. deleteKey(int)

create link Current and equal first
WHILE(Current's data is not equal to the search key)
Current = Current's next
IF(Current equals null)
return null
IF(Current equals first)
first equals Current's next
ELSE
set Current's previous next to Current's next
IF(Current equals last)
last equals Current's previous
ELSE
set Current's next previous to Current's previous
return Current

8. displayForward()

create link Current and equals first
create int nodeCount and equal to 1
Print formatted header
WHILE(Current is not null)
Print Current's data in formatted text
Current equals Current's next
increment nodeCount by one

9. displayBackward()

create link Current and equals last
create in nodeCount and equal to 1
Print formatted header
WHILE(Current is not null)
Print Current's data in formatted text
Current equals Current's previous
increment nodeCount by one

10. displayFirstLink()

return first's data

11. displayLastLink()

return last's data